操作系统课程设计报告

学院：计算机学院

班级：15052316

姓名：鞠畅

学号:15051407

# 实验2 Linux内核模块编译

## 1 实验要求

1.设计一个模块，要求列出系统中所有内核线程的程序名、PID号、进程状态及进程优先级

2.设计一个带参数的模块，其参数为某个进程的PID号，该模块的功能是列出该进程的家族信息，包括父进程、兄弟进程和子进程的程序名、PID号

3.请根据自身情况，进一步阅读分析程序中用到的相关内核函数的源码实现

## 2 核心思路及代码

第一个任务的思路是调用for\_each\_process()函数从0号进程开始，打印出进程的相关信息，通过next\_task找到下一个进程，直至把所有进程都遍历一遍（即next\_task(p)=&init\_task）

代码（初始化函数）

static int info\_init(void)

{

struct task\_struct \*task;

printk(KERN\_ALERT"name\t\tpid\t\tstate\t\tprio\n");

for\_each\_process(task)

{

printk(KERN\_ALERT"%-10s\t%-10d\t%-10ld\t%-10d\n",task->comm,task->pid,task->state,task->prio);

}

return 0;

}

第二个任务的思路是首先声明两个task\_struct的结构体指针，task1指向传入的PID号的进程描述符，如果非空，打印其进程信息。Task2指向该进程的父进程，打印出父进程的信息，之后通过list\_for\_each\_entry从task1的第一个child开始，按sibling遍历，打印出task1所有子进程的信息。然后通过list\_for\_each\_entry，从task1的父进程的第一个子进程开始，按sibling关系遍历，打印出task1所有兄弟进程的信息。

代码（初始化函数）

static int infoByPid\_init(void)

{

struct task\_struct \*task1,\*task2;

task1=pid\_task(find\_get\_pid(pid),PIDTYPE\_PID);

if(task1)

{

printk("self:comm=%s\tpid=%d\tstate=%ld\n",task1->comm,task1->pid,task1->state);

task2=task1->parent;

printk("parent:comm=%s\tpid=%d\tstate=%ld\n",task2->comm,task2->pid,task2->state);

list\_for\_each\_entry(task2,&task1->children,sibling)

{

printk("child:comm=%s\tpid=%d\tstate=%ld\n",task2->comm,task2->pid,task2->state);

}

list\_for\_each\_entry(task2,&task1->real\_parent->children,sibling)

{

if(task2->pid!=pid)

printk("sibling:comm=%s\tpid=%d\tstate=%ld\n",task2->comm,task2->pid,task2->state);

}

}

else

{

printk("process %d doesn't exit!\n",pid);

}

return 0;

}

## 3 心得体会

有第一个实验的基础后，这个实验有思路多了，因为用的大部分函数都和第一个任务相同，但是这个实验当时碰到了一些麻烦，比如受第一个实验的影响，以为find\_get\_pid等内核函数可以直接调用，没有加头文件，编译过程中一直提示函数undefined。按照实验书上引入头文件#include<linux/init.h>，在make的时候总是会包报init\_task undelcared的错，后来通过网上查阅资料，将头文件换成init\_task.h才可以通过。